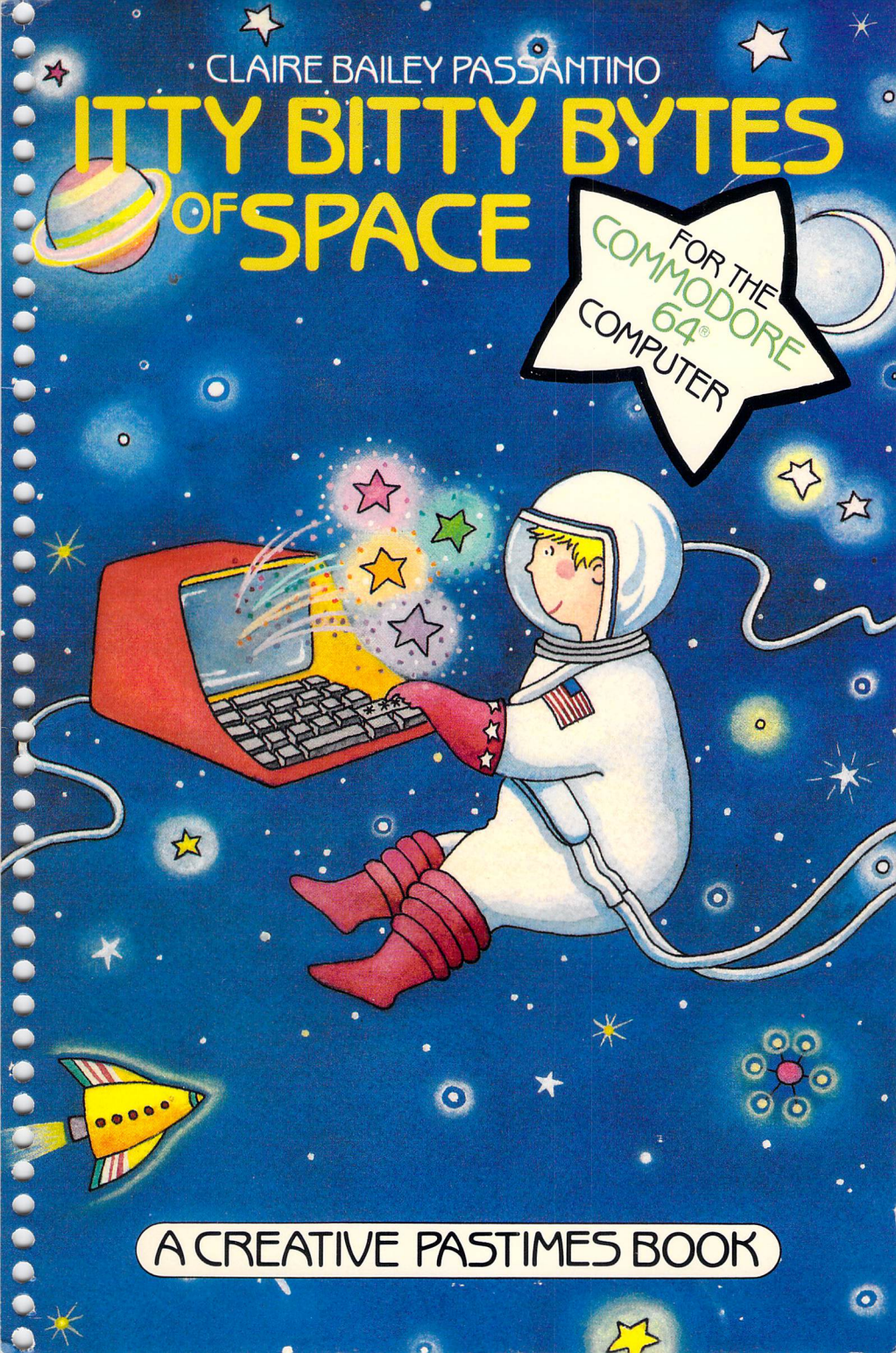
CLAIRE BAILEY PASSANTINO

ITTY BITTY BYTES OF SPACE

FOR THE COMMODORE 64® COMPUTER



A CREATIVE PASTIMES BOOK

This book belongs to

Itty Bitty Bytes of Space

for the Commodore 64® Computer

Claire Bailey Passantino

Text Illustrations by Nancy Gurganus

A Reston Computer Group Book
Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia

A CREATIVE PASTIMES™ BOOK
in the Itty Bitty Bytes Series

Library of Congress Cataloging in Publication Data

Passantino, Claire Bailey.
Itty bitty bytes of space for the Commodore 64 computer.

(A Creative pastimes book)
"A Reston Computer Group book."
1.Commodore 64 (Computer)—Programming—Juvenile literature. 2. Basic (Computer program language)—Juvenile literature. 3. Computer programs—Juvenile literature. 4. Computer games—Juvenile literature.
I. Title. II. Series.
QA76.8.C64P37 1984 001.64'2 84-8420
ISBN 0-8359-3320-2

Cover Illustration by Bethann Thornburgh
Cover design by Nancy Sutherland

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Other Itty Bitty Bytes Books:

Matilda, the Computer Cat
School Days
These titles are available for the
ATARI® computer
TI-99/4A®
and Commodore 64®

Contents

Foreword, vii
A Note to Parents and Teachers, viii
Passing the Test, 2–3
Gravity, 4–5
Test Your Spacecraft, 6–7
Wake Up!, 8–9
Kaboom!, 10–11
Lift Off, 12–13
Music from Mother Earth, 14–15
Count the Stars, 16–17
Make Your Own Kind of Music, 18–19
What Is It?, 20–21
Who Is It?, 22–23
Hit or Miss?, 24–25
Meteors, 26–27
How Much Fuel?, 28–29
Space Sketch, 30–31
Design Your Own Parachute, 32–33
Happy Landing, 34–35
Final Report, 36–37
The Race for Space, 38–39
Break the Code, 40–41
Appendix, 43

Foreword

This is the first of the Itty Bitty Bytes books in the Creative Pastimes series for the Commodore 64 Computer. Each book in the series is designed to bring you twenty fun programs and, at the same time, teach you something about BASIC. Typing in the programs is hard work. Often it will be necessary for you to translate the directions on the printout in order to know what keys to press on the keyboard. You must be careful to number the lines exactly, spell the words correctly, and put in all the required punctuation. But you will feel extra good when your program runs smoothly!

In the Itty Bitty Bytes series of books two pages are devoted to each program. The page on the left contains the program itself. The page on the right contains explanations about each program.

Always read the explanations. Sometimes there are additions or changes that you can make to improve the program. Once you see what the program is all about, feel free to experiment with it—and by all means combine programs that work well together. The more you can do to make a program reflect your own special personality, the more you will enjoy it. Let me know if you find some nifty changes. I love to get mail!

The Itty Bitty Bytes series has grown out of the teaching experiences I've had with my computer students. Special thankyous go to all my "COMPU-KIDS"—and to my own kids, too—who keep presenting me with one good idea after another. They are my severest critics, but my most outstanding assets.

Happy computing!

Claire Bailey Passantino

A Note to Parents and Teachers

You bought the computer. You read the manuals. You did the demo programs. You and/or your children may even have taken some computer classes. But now the computer is just sitting there. Everyone was so enthusiastic. *What happened?*

The novelty of a new computer will take you just so far. Beyond that point, a continuing interest in this incredible tool is directly related to its usefulness. "Not useful" equates with "not used." So the problem becomes, what can computers do that children would find useful? What kinds of things would encourage children to expend the energy needed to create their own computer programs?

Each book in the Itty Bitty Bytes series is packed with computer activities appealing to young programmers. Simple games, contests, races, pictures, designs, songs, riddles, charts, tests, and more—all are designed to be fun while reinforcing beginning computer concepts and skills.

Besides providing fun, there are fringe benefits to having children write their own programs. In working with children, I have found that computer programming encourages them to:

- Think creatively.
- Use logical thinking skills.
- Attend to details.
- Take small steps to achieve a goal.
- Personalize programs.
- Develop pride and self-esteem.
- Appreciate packaged software.

To help children enjoy doing their own programs, here are some suggested DOs and DON'Ts:

- DO encourage children to type in their own programs. With younger children, bargain: "You type this line and I'll type the next one." (Save the long lines for yourself!)
- DON'T criticize typing expertise. Speed and correct fingering are typing skills minimally related to computer programming.
- DO allow children to make mistakes.
- DO help them find the errors they've made. (This is called "debugging" the program.)
- DON'T worry when there is an error message. This means that a mistake has been made. Check the program for "bugs." (The Commodore Computer will give you a ? OUT OF DATA ERROR when you go over a READY signal. Ignore *this* error message. The computer is trying to "READ Y" and there is no "Y" to read!)

- DO encourage children to read and understand the program explanations.
- DON'T, however, force the issue. Some people learn by reading. Others learn by doing. As skills are repeated over and over in different contexts, children may just "catch on."
- DO be aware of some common pitfalls. Remember to
  1. Use line numbers.
  2. Press RETURN after you type in a line of the program, even if the cursor has moved itself to the next line on the screen. Moving the cursor down the screen does not automatically enter the directions into the computer's memory. To do this you must press RETURN.
  3. Give great attention to spelling and punctuation. Quotation marks, commas, semicolons, colons, and even spaces are often very important.
  4. Save your program before you turn off the computer.
- DO help children save their programs on tape or disk so they can use them again and again. If you have a printer, use it to make "hard" copies of each program. People like to see themselves in print.
- DO experiment with the keyboard and get to know how it works in the *immediate* mode. (The immediate mode is when you give the computer a direction without a line number. It responds to your command and does something—*immediately*.) A systematic way to experiment would be to type each key, then type it while pressing the SHIFT, then while pressing CTRL and finally while pressing C=. Example: What happens when you press 3, SHIFT 3, CTRL 3, C= 3?
- DO understand that a *programmed* command may look different from an immediate command, but it will achieve the same results. For example: 10 PRINT "£" is what appears on the screen when you press CTRL 3 inside the quotation marks of a print statement. However, when you type RUN, you will get the same white letters you got by pressing CTRL 3 in the immediate mode.
- DO refer to the Typing Tips in the Appendix. They will help you to understand the printouts of the programs, and how to type them on your own computer. When you are reading the printouts, it is especially important to remember that { } and [ ] and underlined letters are impossible to type on the computer keyboard. This will serve to remind you that these symbols stand for certain typing *procedures*.
- DO remember that RUN/STOP RESTORE gives you back your normal screen.
- DON'T let the multiple commands needed to create sound discourage you and your children from enjoying this feature of the computer. As you become more proficient with sound, the possibilities that these variables afford will intrigue you.

- DO praise children for a job well done. And enjoy the programs that they've created.
- DO modify and use programs that you yourself find useful.

It is my sincere hope that the Itty Bitty Bytes books will help you and your children establish a healthy working relationship with your computer. Take that computer off the shelf! And let me know how things are working out! Good luck.

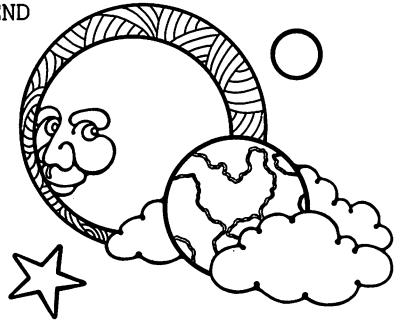
Itty Bitty Bytes of Space

for the Commodore 64® Computer

Passing the Test

How well do you know your planets? To be an astronaut, you must pass this test. A score of 90 is passing. Good luck!

```
1Ø C=Ø:T=Ø
2Ø FOR P=1 TO 5ØØ:NEXT P
3Ø PRINT CHR$(147)
4Ø POKE 53281,Ø
5Ø READ PL$,F$
6Ø IF PL$="END" THEN GOTO 13Ø
7Ø PRINT F$
8Ø PRINT "WHAT IS THE PLANET?"
9Ø INPUT A$
1ØØ T=T+1
11Ø IF A$=PL$ THEN PRINT "RIGHT!":C=C+1:GOTO 2Ø
12Ø IF A$<>PL$ THEN PRINT "WRONG, THE ANSWER
    WAS ";PL$:GOTO 2Ø
13Ø SC=INT(C/T*1ØØ+.5)
14Ø IF SC<9Ø THEN PRINT "SORRY. YOU DID NOT PASS!"
15Ø IF SC>=9Ø THEN PRINT "YOU PASSED THE TEST!"
16Ø PRINT "YOUR SCORE WAS ";SC
2ØØ DATA SATURN,THE PLANET WITH RINGS
21Ø DATA EARTH,OUR PLANET
22Ø DATA VENUS,THE MORNING AND EVENING 'STAR'
23Ø DATA MARS,THE RED PLANET
24Ø DATA JUPITER,THE LARGEST PLANET
25Ø DATA PLUTO,THE FARTHEST PLANET
26Ø DATA MERCURY,NEAREST TO THE SUN
27Ø DATA NEPTUNE,NAMED AFTER THE GOD OF THE SEA
28Ø DATA URANUS,7TH FROM THE SUN
29Ø DATA PLUTO,A FAMOUS DOG
3ØØ DATA END,END
```



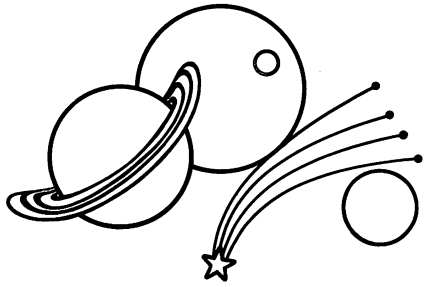
Main Ideas: Designing and scoring a test

The variables used in this program are as follows:

C = Number of correct answers
T = Total number of questions
PL$ = Planet
F$ = Fact about the planet
A$ = Answer
SC = Score

10 At the beginning of the test, there are no correct answers and the total number of questions is 0.
20 Pauses.
30 Clears the screen. Printing CHR$ (147) is the same as pressing SHIFT CLR/HOME.
40 Makes the background color 0 (black).
50 Reads the name of a planet and the fact about it from the data list.
60 Skips to 130 when it reaches the end of the data.
70 Prints the fact about the planet.
80 Asks for your answer.
90 Allows you to input your answer.
100 Adds 1 to the total number of questions.
110 If you answered correctly, adds 1 to the number correct and goes back to line 20 for another question.
120 If you answered incorrectly, tells you the correct planet and goes back for another question.
130 Computes the score.
140–160 Prints an appropriate message.
200–290 The data.
300 Flag data to signal the end of the program. (See line 60.)

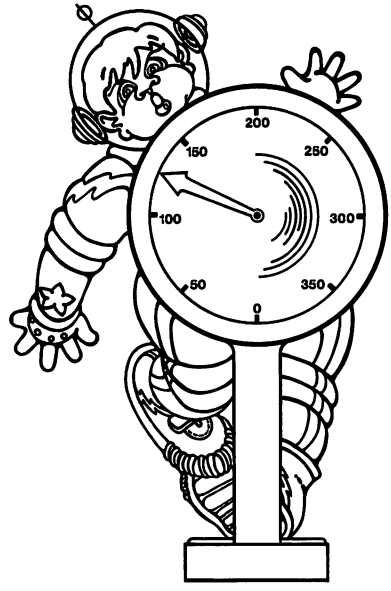
If you know more facts about planets, you can add them to your data list. Make sure the flag data comes last! You may also use this program to make up a test about something else!



Gravity

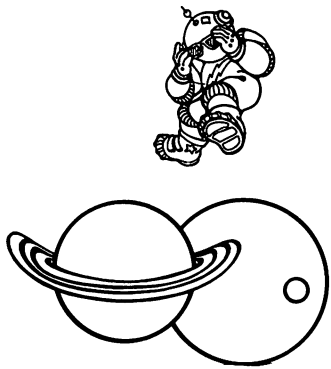
Surface gravity varies on the planets. As part of your training, you must experience various gravitational pulls. How much would you weigh on Pluto? On Venus? How about on Jupiter? Find out!

```
1Ø PRINT CHR$(147)
2Ø PRINT "HOW MUCH DO YOU WEIGH ON EARTH?"
3Ø INPUT W
4Ø PRINT CHR$(147)
45 PRINT
5Ø PRINT "WEIGHT ON EARTH:";W
51 PRINT "{15 C}"
6Ø PRINT "{DOWN}{DOWN}"
7Ø PRINT TAB(7)"WEIGHT ON OTHER PLANETS"
71 PRINT TAB(7)"{23 C}"
8Ø PRINT,"PLANET","WEIGHT"
81 PRINT,"{6 C}","{6 C}"
9Ø PRINT
1ØØ PRINT,"MERCURY",.37*W
11Ø PRINT,"VENUS",.88*W
12Ø PRINT,"MARS",.38*W
13Ø PRINT,"JUPITER",2.64*W
14Ø PRINT,"SATURN",1.15*W
15Ø PRINT,"URANUS",.79*W
16Ø PRINT,"NEPTUNE",1.12*W
17Ø PRINT,"PLUTO",.Ø3*W
```



Main Idea: Printing a chart

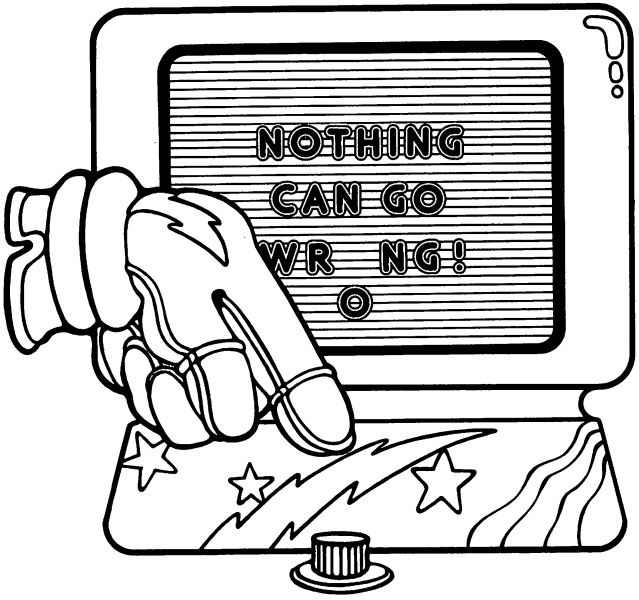
10 Clears the screen.
20 Asks the question.
30 Allows you to input your weight.
40 Clears the screen again.
45 Skips a line.
50 Prints your weight on earth.
51 Underlines. I used SHIFT C to underline, but you can use any character you like. Experiment!
60 Moves the cursor down 2 rows. To enter this line, press the ↑ CRSR ↓ key two times. It will look like a Q on the screen.
70 Counts over to column 7 and prints the title.
71 Counts over to column 7 and underlines the title.
80 The first comma moves the cursor to the second print zone, where the column heading is printed. The second comma moves to the third print zone, where the next column heading is printed.
81 Underlines the column headings.
90 Skips a line.
100–170 For each planet, prints the name of the planet, then computes and prints your weight on that planet.



Test Your Spacecraft

Tomorrow you will take off into outer space. Today you must test your spacecraft. Don't worry. Nothing can go wrong!

```
1Ø PRINT CHR$(147)
2Ø PRINT "WHAT IS YOUR NAME?"
3Ø INPUT N$
4Ø PRINT CHR$(147)
5Ø POKE 53281,Ø
6Ø POKE 5328Ø,2
7Ø FOR R=1 TO 1Ø:PRINT:NEXT R
8Ø PRINT TAB(5);"HELLO CAPTAIN ";N$
9Ø PRINT
1ØØ PRINT TAB(5);"ARE YOU READY TO GO?";
11Ø INPUT A$
12Ø PRINT
13Ø PRINT TAB(8);"NOTHING CAN GO WRONG"
14Ø PRINT
15Ø FOR P=1 TO 15ØØ:NEXT P
16Ø PRINT "WRONG";
17Ø BG=INT(16*RND(1))
18Ø POKE 53281,BG
19Ø GOTO 16Ø
```



Main Idea: Holding a conversation with the computer
An infinite loop

10 Clears the screen. PRINT CHR$(147) works the same way as PRINT "♥" which is obtained by pressing SHIFT CLR/HOME. You may use these statements interchangeably.
20 Asks the question.
30 Allows you to tell your name.
40 Clears the screen.
50 Makes the background color 0 (black).
60 Makes the border color 2 (red).
70 Prints 10 blank lines.
80 Goes to column 5 to print the message.
90 Skips a line by printing a black line.
100 Goes to column 5 to ask the question.
110 Allows you to answer the question.
120 Skips a line.
130 Goes to column 8 to print the message.
140 Skips a line.
150 Pauses while the computer counts to 1500.
160–190 An infinite loop. Press RUN/STOP to break the program. Press RUN/STOP RESTORE to get back to the normal screen.
160 Prints "WRONG." Try a comma instead of the semicolon. Then take out the comma entirely.
170 Selects a number from 0 to 15 for the background color.
180 Changes the background color. To change the border color instead, type:

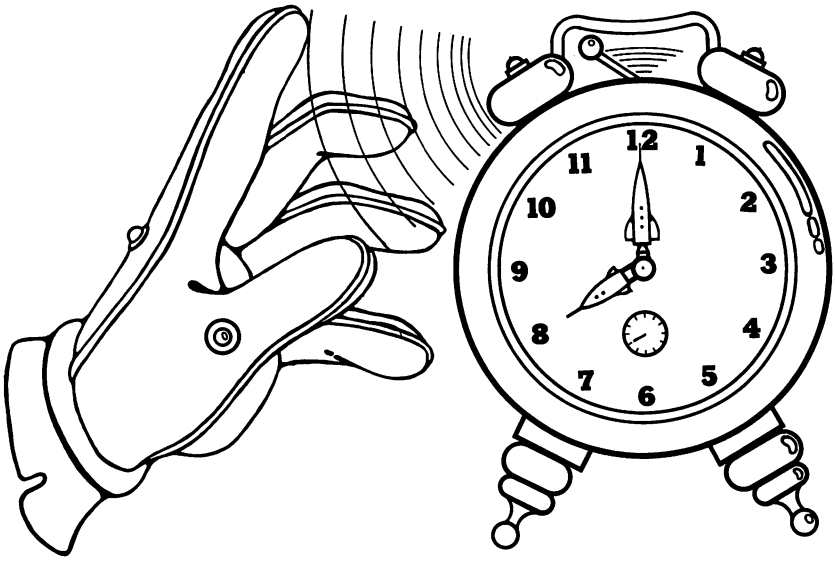
```
180 POKE 53280, BG
```

190 Goes back to print the repeating message on a flashing screen.

Wake Up!

Adjustments have been made to your spacecraft. Better get some sleep. Tomorrow's your big day. Don't forget to set the alarm clock.

```
1Ø PRINT "{CLR} {10 DOWN}"
2Ø POKE 53281,Ø
3Ø PRINT "{WHT}"SPC(12) "NIGHTY NIGHT!"
4Ø FOR P=1 TO 3ØØØ:NEXT P
5Ø FOR R=54272 TO 54296:POKE R,Ø:NEXT R
6Ø POKE 53281,7
7Ø PRINT "{CLR} {1Ø DOWN}"
8Ø PRINT "{RED}"SPC(13)"WAKE UP!"
9Ø POKE 54273,6Ø
1ØØ POKE 54277,9
11Ø POKE 54296,15
12Ø FOR N=1 TO 12
13Ø POKE 54276,21
14Ø FOR P=1 TO 7Ø:NEXT P
15Ø POKE 54276,2Ø
16Ø NEXT N
17Ø POKE 53281,2
18Ø FOR P=1 TO 5Ø:NEXT P
19Ø GOTO 6Ø
```



Main Idea: Making an alarm clock sound

10 Clears the screen and moves the cursor down 10 rows. Type this line by pressing SHIFT CLR/HOME, then 10 ↑ CRSR ↓.

20 Makes the screen black.

30 Makes the letters white, skips 12 spaces, and prints. To get the {WHT}, type CTRL 2. It will look like E on the screen.

40 Pauses while the computer counts to 3000.

50 Clears all the sound registers. The addresses devoted to sound are numbered 54272 to 54296.

60 Makes the screen yellow.

70 Clears the screen and moves the cursor down 10 rows. (See line 10 for directions on how to type.)

80 Makes the letters red, skips 13 spaces, and prints. To get the {RED}, type CTRL 3. It will look like £ on the screen. (On the yellow screen, the red letters look almost black.)

90 Sets the high frequency at 60. (For a different pitch, you may select numbers from 1 to 255.)

100 Sets the variable for attack/decay.

110 Sets the loudness (15 is loudest).

120–160 This section makes one trill. To make the trill, the waveform is turned on (line 130), held while the computer counts to 70 (line 140), and then turned off (line 150). This procedure is repeated 12 times.

170 Makes the background red. The letters seem to disappear.

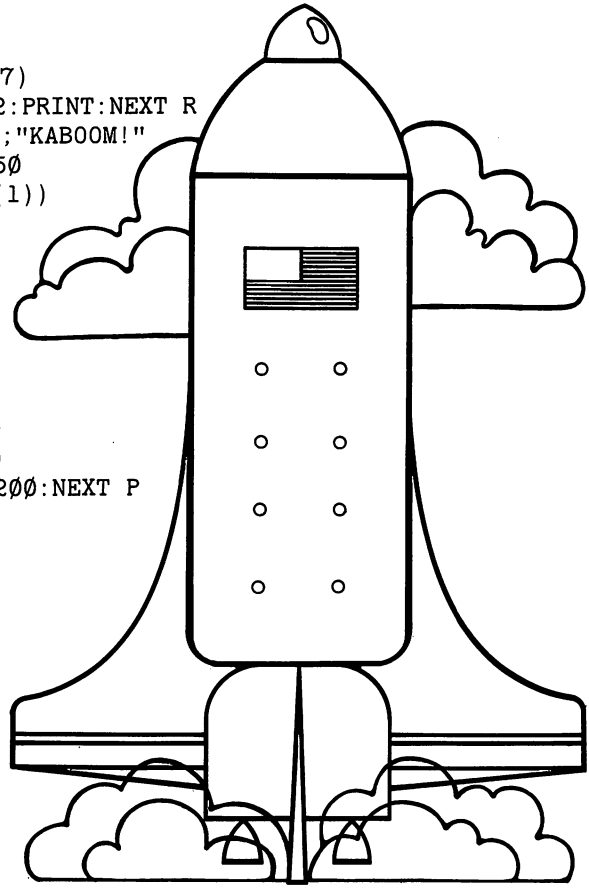
180 Pauses briefly while the computer counts to 50.

190 Goes back to line 60 to tell you to wake up again. This procedure will repeat until you turn off the "alarm." To do so, press RUN/STOP. To return to the blue screen, press RUN/STOP RESTORE.

Kaboom!

All systems are go. Blastoff!

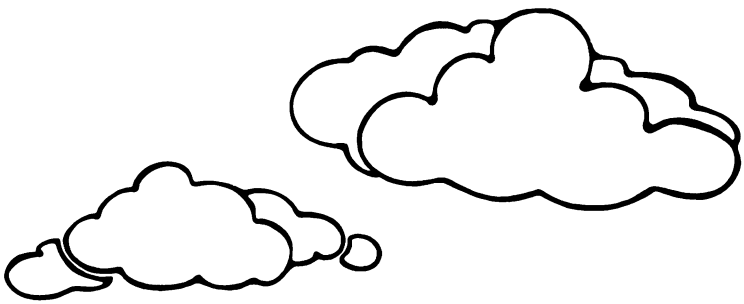
```
1Ø FOR L=54272 TO 54296:POKE L,Ø:NEXT L
2Ø V=54296
21 AT=54277
22 SR=54278
23 HF=54273
24 LF=54272
25 WF=54276
3Ø PRINT CHR$(147)
4Ø FOR R=1 TO 12:PRINT:NEXT R
5Ø PRINT TAB(17);"KABOOM!"
6Ø FOR FL=1 TO 5Ø
7Ø C=INT(16*RND(1))
8Ø POKE 53281,C
9Ø POKE V,15
91 POKE WF,129
92 POKE AT,15
93 POKE HF,4Ø
94 POKE LF,2ØØ
1ØØ NEXT FL
11Ø POKE 53281,Ø
12Ø POKE 5328Ø,Ø
13Ø FOR P=1 TO 2ØØ:NEXT P
```



(Add the next program to this one!)

Main Ideas: Flashing colors
Using RESTORE to enable reuse of data

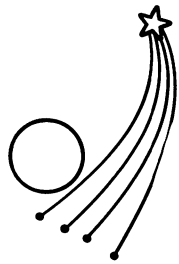
10 Clears all the sound registers.
20–25 Assigns variable names for the appropriate addresses. V stands for volume, AT for attack/decay, SR for sustain/release, HF and LF for high and low frequency, WF for waveform.
30 Clears the screen.
40 Prints 12 blank rows.
50 Prints "KABOOM!" near the center of the screen.
60 The blastoff will consist of 50 flashes.
70 Selects a number from 0 to 15 for color.
80 Makes the screen whatever color was selected.
90–94 Pokes values into sound addresses. 15 is the loudest volume, 129 is the waveform for noise. The other values were experimental. Try your own!
100 Goes back to line 60 until the 50 flashes are completed.
110 Makes the background black.
120 Makes the border black.
130 Pauses.



Lift Off

It's up, up and away!

```
15Ø PRINT "{CLR}{21 DOWN}"
16Ø PRINT TAB(19);"Z {4 SPACES}UP UP..."
17Ø PRINT TAB(18);"N {1 SPACE} M"
18Ø PRINT TAB(18);"{G} {1 SPACE} {M}"
19Ø PRINT TAB(17);"U {3 {P}} I"
2ØØ FOR LOUD=12 TO Ø STEP -1
21Ø POKE V,LOUD
22Ø POKE AT,96
23Ø POKE SR,Ø
24Ø POKE WF,17
25Ø READ H,L
26Ø IF H=-1 THEN 3ØØ
27Ø POKE HF,H:POKE LF,L
28Ø POKE WF,16
29Ø GOTO 21Ø
3ØØ PRINT:PRINT
31Ø FOR T=1 TO 3Ø:NEXT T
32Ø RESTORE
33Ø NEXT LOUD
34Ø PRINT TAB(27);"...AND AWAY!"
35Ø GOTO 35Ø
4ØØ DATA 33,135,35,134,37,162,39,223,42,62
41Ø DATA 44,193,47,1Ø7,5Ø,6Ø,53,57,56,99
42Ø DATA 59,19Ø,63,75,67,15
43Ø DATA -1,-1
```



Main Idea: Moving an image up the screen

You may run this program by itself, but it works well as an addition to the last program.

150 Moves the cursor home, then down 21 rows. To get the {CLR}, press SHIFT CLR/HOME. It will look like ♥ on the screen. To get the {21 DOWN}, press the ↑CRSR↓ 21 times. It will look like QQQQQQQQQQQQQQQQQQQQQ

160–190 Draws the spaceship. Here are some helpful hints:

160 Press SHIFT Z. It will look like "♦".

170 Press SHIFT N, space, SHIFT M. It will look like "/\".

180 Press C= G, space, C= M. It will look like "| |".

190 Press SHIFT U, 3 C= P, SHIFT I. It will look like "(——)".

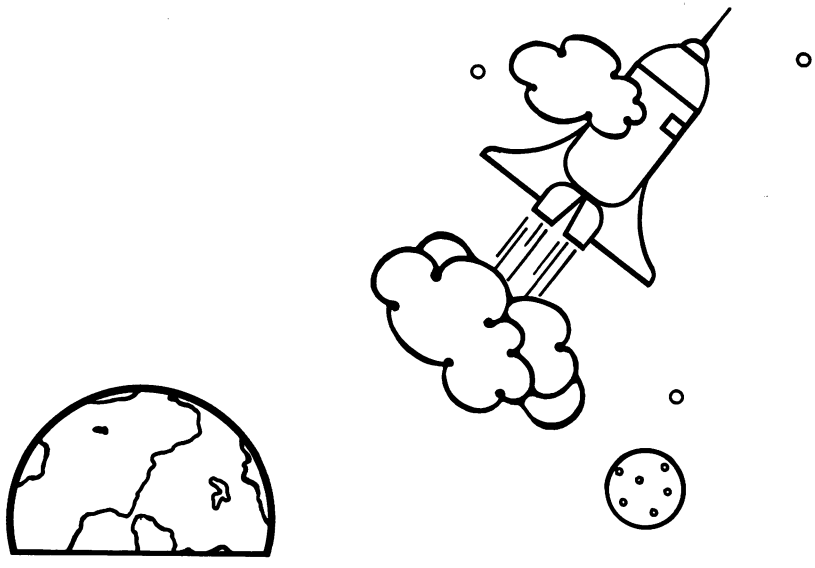
200–330 Makes 12 sets of lift-off sounds. Each set is a little softer than the last, and between each set 2 blank lines are printed (line 300) to give the illusion of moving the spaceship up the screen.

340 Print this when finished.

350 A dummy line to freeze the message and prevent the READY signal. It is called a dummy line because it keeps going to itself. Pretty dumb!

400–420 Data for one set of sounds. The data is restored by line 320 so it can be used for all 12 blips.

430 This is flag data used in line 260 to signal the end of one blip and move the spaceship up.



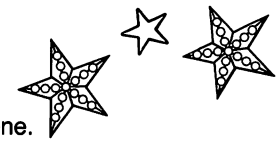
Music from Mother Earth

What's this? They're sending you music from Earth. They must be testing the sound system!

```
1Ø PRINT CHR$(147)
2Ø POKE 53281,Ø
3Ø PRINT CHR$(5)
4Ø FOR R=1 TO 8:PRINT:NEXT R
5Ø PRINT TAB(5);"TWINKLE TWINKLE LITTLE STAR"
6Ø PRINT
7Ø PRINT TAB(3);"HOW I WONDER HOW MANY THERE ARE!"
8Ø FOR L=54272 TO 54296:POKE L,Ø:NEXT L
9Ø V=54296:WF=54276 :AT=54277 :HF=54273 :LF=54272 :
   SR=54278 : HP=54275 : LP=54274
17Ø POKE V,15
18Ø POKE AT,88
19Ø POKE SR,89
2ØØ POKE HP,9
21Ø POKE LP,Ø
22Ø READ H,L,T
23Ø IF H=-1 THEN 4ØØ
24Ø POKE HF,H
25Ø POKE LF,L
26Ø POKE WF,65
27Ø PRINT "{HOME}"
28Ø PRINT:PRINT TAB(5);"*"
29Ø FOR P=1 TO 2ØØ*T:NEXT P
3ØØ POKE WF,64
31Ø PRINT CHR$(144)
32Ø PRINT "{HOME}"
33Ø PRINT:PRINT TAB(5);"*"
34Ø FOR P=1 TO 3Ø:NEXT P
35Ø PRINT CHR$(5)
36Ø GOTO 17Ø
37Ø DATA 33,135,1,33,135,1,5Ø,6Ø,1,5Ø,6Ø,1,56,99,1,56,
    99,1,5Ø,6Ø,2
38Ø DATA 44,193,1,44,193,1,42,62,1,42,62,1,37,162,1,37,
    162,1,33,135,2
39Ø DATA -1,-1,-1
4ØØ GOTO 4ØØ
```

Main Ideas: Playing a song
Making an image blink

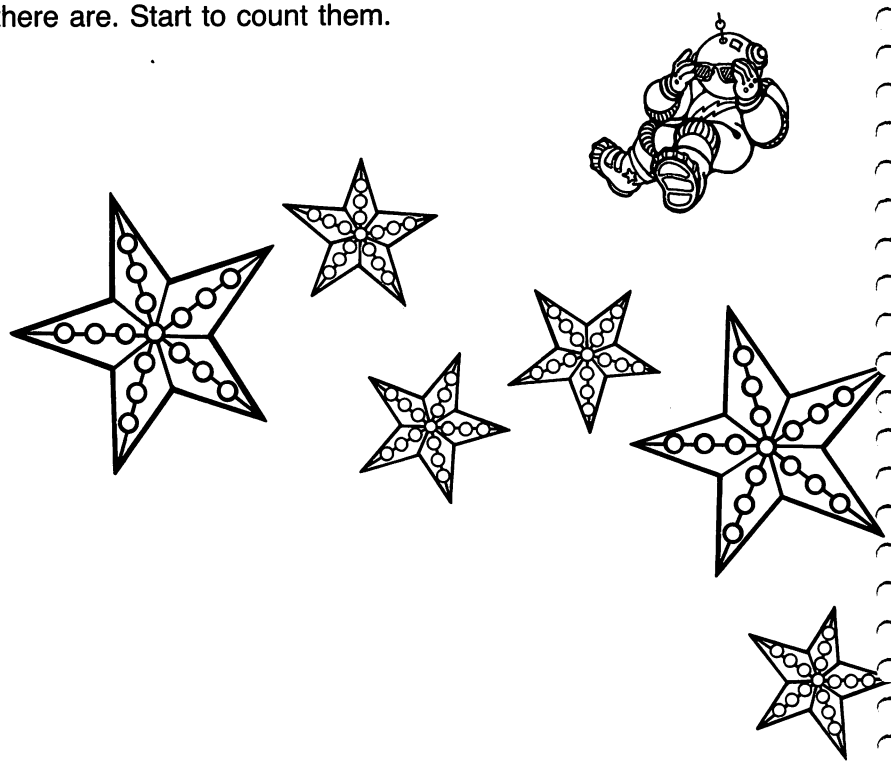
10 Clears the screen.
20 Makes the screen black.
30 Characters will be printed in white.
40 Prints 8 blank lines.
50–70 Positions and prints the title.
80 Clears all the locations for sound.
90 Assigns variable names to locations dealing with sound. These locations are: volume (54296), waveform (54276), attack/decay (54277), high frequency (54273), low frequency (54272), sustain/release (54278), high pulse (54275), low pulse (54274).
170–210 Assigns values to various sound variables. Changing these values will produce different kinds of sounds.
220 Reads three numbers from the data (in lines 370–390). H and L are the high and low frequencies for the note. T is the time (number of beats). Refer to the chart of Musical Notes in the Appendix.
230 Checks for "flag" data. When the computer reads a −1 for the H value, the program skips to line 400.
240–260 Assigns the first two values read in line 220 to high frequency and low frequency. Sets waveform to pulse (65).
270 Moves the cursor to the home position in the upper left corner. Press CLR/HOME. It will look like S.
280 Prints a black line, then counts to column 5 and prints a star (in white).
290 Uses the third value read in line 220 to determine how many beats to hold each note. To slow down the song, pick a number larger than 200 for the length of one beat. To speed it up, choose a smaller number.
300 Turns off the waveform.
310 Sets up black as the next color to use for printing characters.
320 Sends the cursor home.
330 Prints a black star on top of the white star, thereby erasing it.
340 Pauses.
350 Goes back to white, for printing characters. By switching from black to white to black, etc., the star appears to blink.
360 Goes back to 170 to play the next note and print a blinking star.
370–380 Data for each note.
390 "Flag" data used to signal the end of the song. (See line 230.)
400 Freezes the image on the screen.



To count the stars, add the next program to this one.

Count the Stars

The stars are just beautiful! It's amazing how many of them there are. Start to count them.

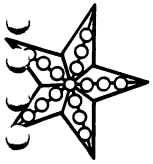


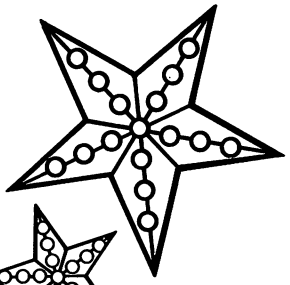
```
40Ø REM COUNT STARS
41Ø CT=Ø
42Ø FOR R=1 TO 2Ø:PRINT:NEXT R
43Ø PRINT "   PRESS ANY KEY TO COUNT THE STARS"
44Ø GET K$:IF K$="" THEN 44Ø
45Ø PRINT CHR$(147)
46Ø CT=CT+1
47Ø L=INT(4Ø*RND(1))
48Ø PRINT CT;TAB(L);"*"
49Ø GOTO 46Ø
```

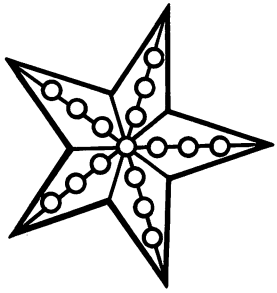
Main Ideas: Incrementing a variable
An infinite loop

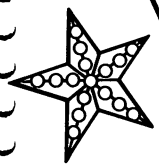
For extra fun, add this program to your last program!

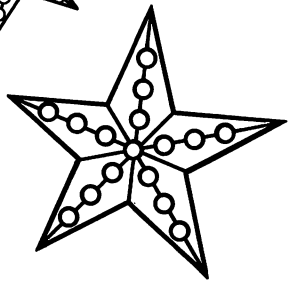
400 A remark.
410 Sets the counter at 0.
420 Prints 20 blank rows.
430 Prints directions near the bottom of the screen.
440 Waits until a key is pressed.
450 Clears the screen.
460 Increments the count. This means that the count increases. In this case, the count is incremented by one.
470 Picks a random number from 0 to 39.
480 Prints the count and then prints a star at the random location.
490 Goes back to 460 to print the next star, and the next, and the next, and . . . this loop will continue forever. It is called an infinite loop. To break the loop, press **RUN/STOP**. To return to the normal screen, press **RUN/STOP** **RESTORE**.

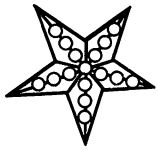








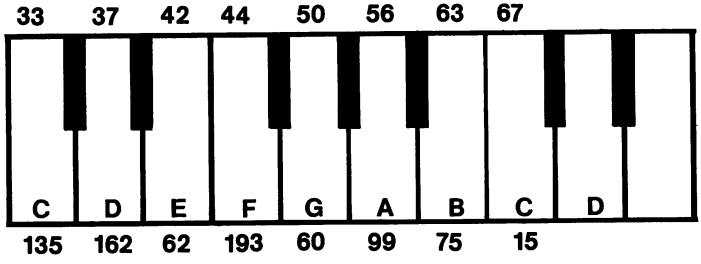




Make Your Own Kind of Music

Why not play your own song? Press keys 1 to 8!

```
1Ø PRINT CHR$(147)
2Ø FOR R=1 TO 9:PRINT:NEXT R
3Ø PRINT TAB(4);"TO MAKE MUSIC, PRESS KEYS 1-8."
4Ø K=Ø:POKE 54276,64
6Ø POKE 54296,15:REM VOLUME
7Ø POKE 54277,2Ø:REM ATTACK
8Ø POKE 54278,89:REM SUSTAIN-RELEASE
9Ø POKE 54275,9:REM HIGH PULSE
1ØØ POKE 54274,Ø:REM LOW PULSE
11Ø GET K:IF K<1 OR K>8 THEN 11Ø
12Ø POKE 54276,65:REM WAVEFORM
```




```
13Ø IF K=1 THEN POKE 54273,33:POKE 54272,135
14Ø IF K=2 THEN POKE 54273,37:POKE 54272,162
15Ø IF K=3 THEN POKE 54273,42:POKE 54272,62
16Ø IF K=4 THEN POKE 54273,44:POKE 54272,193
17Ø IF K=5 THEN POKE 54273,5Ø:POKE 54272,6Ø
18Ø IF K=6 THEN POKE 54273,56:POKE 54272,99
19Ø IF K=7 THEN POKE 54273,63:POKE 54272,75
2ØØ IF K=8 THEN POKE 54273,67:POKE 54272,15
21Ø GOTO 4Ø
```

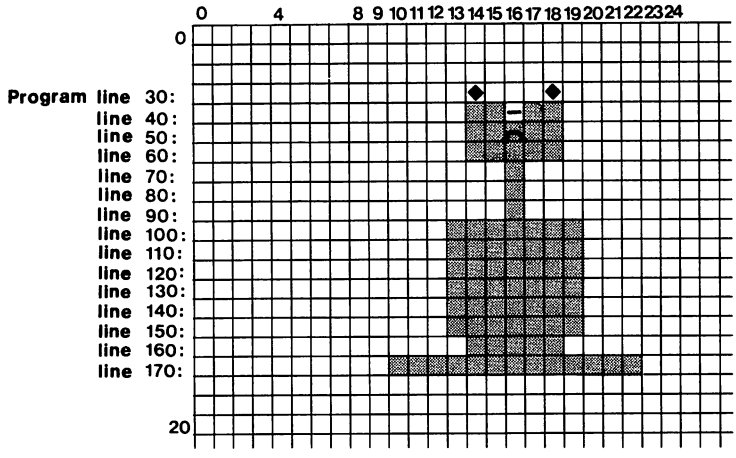
Main Ideas: Using the keyboard to play songs

10 Clears the screen.
20 Skips 9 lines by printing 9 blank rows.
30 Goes to column 4 and prints directions.
40 Sets K to 0 and turns off the waveform.
60–100 Sets the variables to make sound. Changing these variables changes the characteristics of the sound.
100 Sets up the keyboard for input. When you press a key, the value of K changes.
120 Turns on the waveform.
130–200 Depending on the key pressed, the high and low frequencies are set. Then the program goes back to line 40 to turn off the note and get the next one. For a complete chart of the notes and their high and low frequency values, refer to the Appendix.

What Is It?

Something scary has appeared on your screen. What is it?

```
1Ø PRINT CHR$(147)
2Ø PRINT:PRINT
3Ø GOSUB 19Ø:PRINT SPC(4)"Z"SPC(3)"Z"
4Ø GOSUB 19Ø:PRINT SPC(4)"{2 SPACES}{*}{2 SPACES}"
5Ø GOSUB 19Ø:PRINT SPC(4)"{5 SPACES}"
6Ø GOSUB 19Ø:PRINT SPC(4)"{SPACE} N {T} M [SPACE]"
7Ø GOSUB 19Ø:PRINT SPC(6)"{SPACE}"
8Ø GOSUB 19Ø:PRINT SPC(6)"{SPACE}"
9Ø GOSUB 19Ø:PRINT SPC(6)"{SPACE}"
1ØØ GOSUB 19Ø:PRINT SPC(3)"{7 SPACES}"
11Ø GOSUB 19Ø:PRINT SPC(3)"{7 SPACES}"
12Ø GOSUB 19Ø:PRINT SPC(3)"{M} {5 SPACES} {G}"
13Ø GOSUB 19Ø:PRINT SPC(3)"{M} {5 SPACES} {G}"
14Ø GOSUB 19Ø:PRINT SPC(3)"{M} {5 SPACES} {G}"
15Ø GOSUB 19Ø:PRINT SPC(3)"{M} {5 SPACES} {G}"
16Ø GOSUB 19Ø:PRINT SPC(4)"{5 SPACES}"
17Ø GOSUB 19Ø:PRINT "{6 SPACES} B {6 SPACES}"
18Ø GOTO 3ØØ
19Ø REM GOSUB ROUTINE
2ØØ PRINT TAB(12);
21Ø PRINT CHR$(18);
22Ø PRINT CHR$(3Ø);
23Ø RETURN
3ØØ REM END
```



Main Ideas: Drawing a picture on the screen
Utilizing a GOSUB routine

10 Clears the screen.
20 Prints 2 blank lines.

30–170 Draws the creature, using a GOSUB routine to repeat procedures. Remember that underlined letters in the printout are made by typing SHIFT with the letter. Letters in { } are made by typing C= with the letter. SPC instructs the computer to skip over a certain number of spaces.
30 Prints two ♦ shapes for eyes. Since the GOSUB turn on reversed characters and green color, the eyes appear on a green background.
40 Prints 2 green spaces, a — for the nose, and another 2 spaces.
50 Skips over 4 spaces and prints 5 green spaces.
60 Prints a mouth ⌒.
70–90 Each line skips over 6 spaces and prints a space for the neck.
100–110 Both lines skip over 3 spaces and print 7 spaces.
120–150 The C= M and the C= G make the dark lines that separate the arms from the 5 spaces of body.
160 Skips over 4 spaces and prints 5 spaces.
170 Prints 6 spaces for each foot. SHIFT B prints the line between the feet.

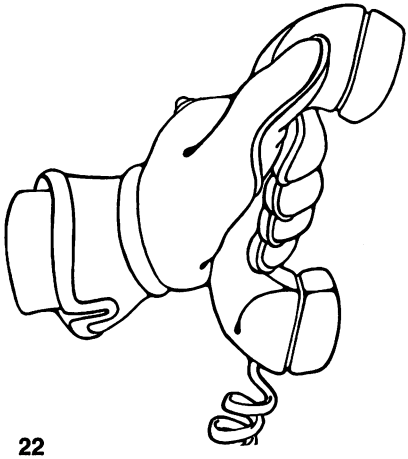
180 When finished drawing, skips to 300.
190–230 The GOSUB routine:
200 Goes to column 12.
210 Turns on reverse.
220 Selects green color.
230 Returns to the program.
300 Ends.

To find out who this creature is, add the next program to this one.

Who Is It?

Aha! Do you know who it is? Add these lines to your last program to solve the mystery!

```
3ØØ REM TEE HEE MUSIC
31Ø FOR L=54272 TO 54296:POKE L,Ø:NEXT L
32Ø POKE 54296,15:REM LOUDNESS
33Ø POKE 54277,88:REM ATTACK
34Ø POKE 54278,89:REM SUSTAIN
35Ø POKE 54275,9:REM HIGH PULSE
36Ø POKE 54274,Ø:REM LOW PULSE
37Ø READ H,L,T
38Ø IF H=-1 THEN 49Ø
39Ø C=INT(16*RND(1))
4ØØ POKE 53281,C
41Ø POKE 54273,H:POKE 54272,L:REM HIGH/LOW FREQUENCY
42Ø POKE 54276,65:REM WAVEFORM
43Ø FOR P=1 TO 1ØØ*T:NEXT P
44Ø POKE 54276,64:REM WAVEFORM
45Ø GOTO 37Ø
46Ø DATA 33,135,3,5Ø,6Ø,6,44,193,1,42,62,1,37,162,1,
    33,135,1,31,165,3,33,135,6
47Ø DATA 28,49,3,56,99,6,5Ø,6Ø,1,44,193,1,42,62,1,37,
    162,1,35,134,3,37,162,6
48Ø DATA -1,-1,-1
49Ø PRINT "{HOME}":FOR R=1 TO 2Ø:PRINT:NEXT R
5ØØ PRINT "WATCH OUT FOR LASER BEAMS!"
51Ø PRINT "IT'S TEE HEE, UNFRIENDLY ALIEN WHO NEVER
    EVER PHONES HOME!"
```

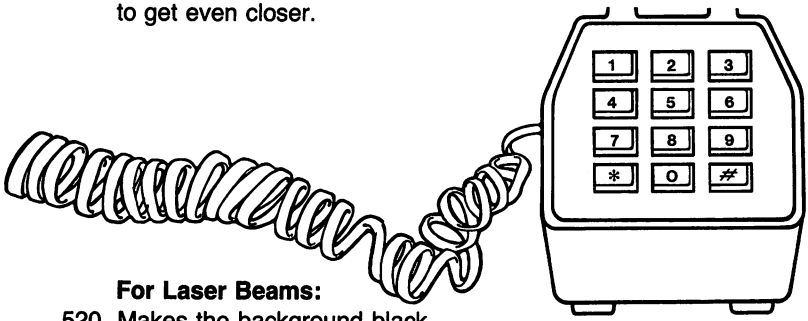


Watch out for laser beams!

```
52Ø POKE 53281,Ø
53Ø PRINT "{HOME}":PRINT
54Ø PRINT TAB(12) CHR$(144);
55Ø PRINT SPC(4)"*"SPC(3)"*"
56Ø FOR P=1 TO 3Ø:NEXT P
57Ø PRINT "{HOME}":PRINT
58Ø PRINT TAB(12) CHR$(5);
59Ø PRINT SPC(4)"*"SPC(3)"*"
6ØØ GOTO 53Ø
```

Main Ideas: Combining graphics and sound
Flashing background
Making an object appear, disappear and reappear

300 Remark.
310–360 Pokes values into locations for sound.
370 Reads high and low frequency, and time for each note.
380 Checks for flag data. Flag data is used to signal the computer. In this case, it will go to line 490.
390 Picks a random color.
400 Changes the background to the color picked.
410 Pokes in the values for high and low frequency.
420 Turns on the waveform.
430 Holds the note 100 units for each beat.
440 Turns off the waveform.
450 Goes back to 370 to set up the next note.
460–470 Data for the notes.
480 Flag data.
490 Brings the cursor home, then counts down 20 rows before printing. When you press CLR/HOME it looks like "S" on the screen.
500–510 Prints the message.
515 Moves the drawing up closer to the laser beams. Try PRINT:PRINT to get even closer.

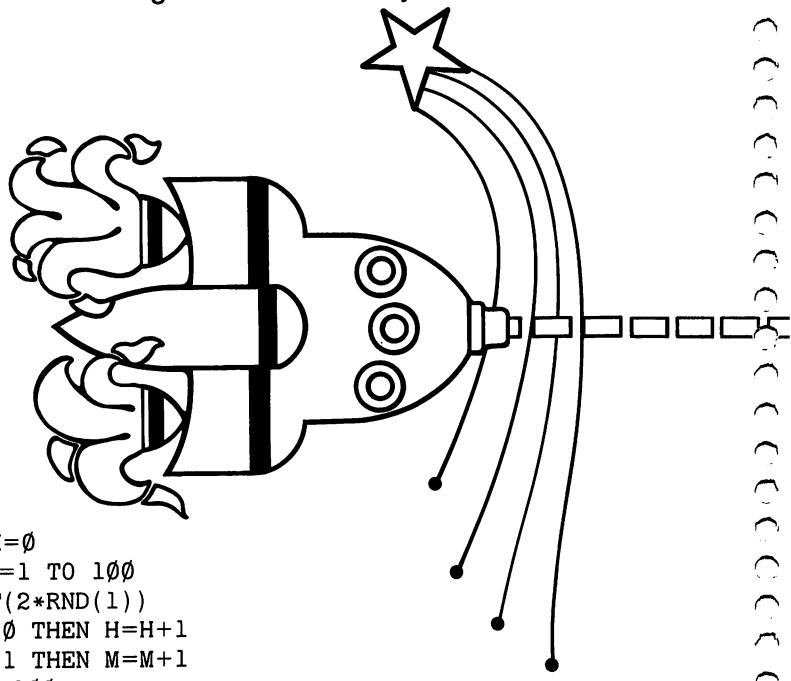


For Laser Beams:

520 Makes the background black.
530 Brings the cursor home and skips a line.
540 Goes to column 12 and selects black as the drawing color.
550 Skips 4 spaces, prints a black star, skips 3 spaces and prints another black star (invisible since the background is black).
560 Pauses and counts to 30. Change the 30 to a larger number for a longer pause.
570 Brings the cursor home and skips a line.
580 Goes to column 12 and selects white as the drawing color.
590 Draws 2 white stars where the black ones had been drawn, flashing them on the screen.
600 Goes to 530 to draw black stars thereby erasing the white ones. Changing from black to white to black again gives the illusion of blinking.

Hit or Miss?

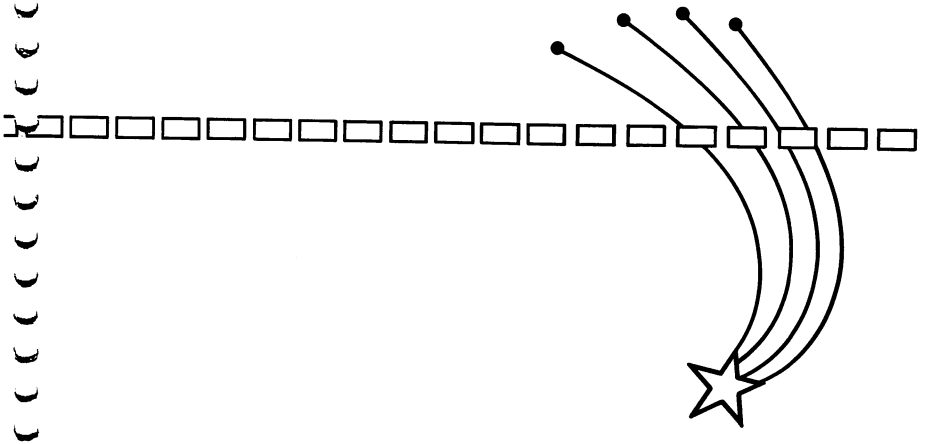
You have 100 rounds of ammunition. Fifty hits will knock him out, or at least damage his lasers. Can you do it?



```
1Ø H=Ø:M=Ø
2Ø FOR R=1 TO 1ØØ
3Ø X=INT(2*RND(1))
4Ø IF X=Ø THEN H=H+1
5Ø IF X=1 THEN M=M+1
6Ø GOSUB 2ØØ
7Ø POKE 54276,128
8Ø PRINT "HITS ";H,"MISSES ";M
9Ø NEXT R
1ØØ IF H<5Ø THEN PRINT "NO MORE CALLS TO GRANDMA!":END
11Ø PRINT "YOU DID IT!"
12Ø PRINT "PHONE HOME AND TELL THEM ALL ABOUT IT!":END
2ØØ REM SHOTS
21Ø FOR V=15 TO 1Ø*RND(1) STEP -1
22Ø POKE 54296,V:REM VOLUME
23Ø POKE 54276,129:REM WAVEFORM NOISE
24Ø POKE 54277,15:REM ATTACK
25Ø POKE 54273,255*RND(1):REM HIGH FREQUENCY
26Ø POKE 54272,255*RND(1):REM LOW FREQUENCY
27Ø NEXT V
28Ø RETURN
```

Main Ideas: Random outcome
Initializing and incrementing variables
Using a GOSUB routine

10 Initializes the variables at 0. H stands for hits and M stands for misses.
20–90 Loops through 100 rounds of ammunition.
30 Selects a 0 or a 1.
40 If 0 was selected, adds one to the number of hits.
50 If 1 was selected, adds one to the number of misses.

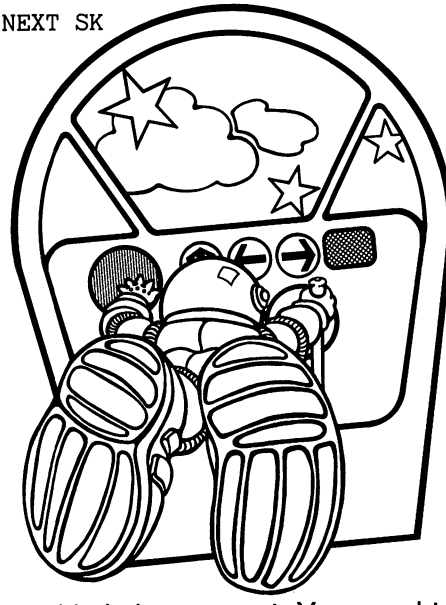


60 Goes to line 200 and makes a shooting sound.
70 Turns off the sound of the shot.
80 Prints the current score.
90 Returns to line 20 for another round of ammunition.
100 Prints this line if the number of hits is less than 50.
100–120 Prints this if you had at least 50 hits.
200–280 This is the GOSUB routine to produce shooting noises.
210 Reduces the volume from loudest (15) to some number chosen at random, from 0 to 9.
220–260 Sets the sound variables. The high and low frequencies are selected at random. Perform your own experiments and pick your own numbers (from 0 to 255).
270 Loops back to line 210 for as many times as needed to reduce the volume.
280 Returns to the program.

Meteors

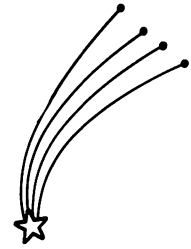
It's a meteor shower. Can you steer your way through it? Good luck!

```
1Ø PRINT "{CLR}"
2Ø FOR SK=1 TO 25:PRINT:NEXT SK
3Ø POKE 5328Ø,5
4Ø POKE 53281,Ø
5Ø L=2Ø
6Ø POKE 1824+L,81
7Ø POKE 56Ø96+L,5
8Ø R=INT(4Ø*RND(1))
9Ø POKE 1984+R,16Ø
1ØØ POKE 56256+R,6
11Ø J=PEEK(5632Ø)
12Ø IF J=117
    OR J=118 OR J=119
    THEN L=L+1
13Ø IF J=121 OR
    J=122 OR J=123
    THEN L=L-1
14Ø IF L>39 THEN L=39
15Ø IF L<Ø THEN L=Ø
16Ø PRINT
17Ø GOTO 6Ø
```



How about some collisions! And let's keep count. You need to assess the damages to your spacecraft.

```
45 T=Ø:C=Ø
55 IF PEEK(1824+L)<>32 THEN GOSUB 3ØØ
162 T=T+1
165 IF T=2ØØ THEN GOTO 4ØØ
17Ø GOTO 55
3ØØ REM COLLISION
31Ø FOR E=1 TO 2Ø
32Ø F=INT(16*RND(1))
33Ø POKE 53281,F
34Ø NEXT E
35Ø POKE 53281,Ø
36Ø C=C+1
37Ø RETURN
4ØØ REM ENDING
41Ø PRINT "THAT WAS ";C;" COLLISIONS!"
```



Main Ideas: Using the joystick to plot a course down the screen

10 Clears the screen. To type the {CLR}, press SHIFT CLR/HOME. It looks like ♥ on the screen.
20 Prints 25 blank lines. This moves the cursor down to the bottom of the screen. The next time something is printed, the screen will scroll up. This scrolling action gives the illusion of motion.
30 Makes the border color 5 (green).
40 Makes the background color 0 (black).
50 Your beginning location is 20, halfway across the 40 column screen.
60–70 Refer to your Screen Memory Map and Color Memory Map in the Appendix. Line 1824 and line 56096 are each 5 rows from the bottom. Display Code 81 (a circle) will be drawn for you in color 5 (green).
80 Picks a random integer from 0–39.
90–100 Displays code 160 (a reversed space) at a random spot in the bottom row. This stands for a meteor and will be drawn in color 6 (blue).
110 Checks the joystick position.
120 If you've pushed towards the right, 1 is added to your location (L).
130 If you've pushed towards the left, 1 is subtracted.
140–150 Keeps you at the edge of the screen if you push too far right or left.
160 Prints a blank line, thereby making the screen scroll upwards.
170 Goes back to line 60 to draw you and the next meteor.

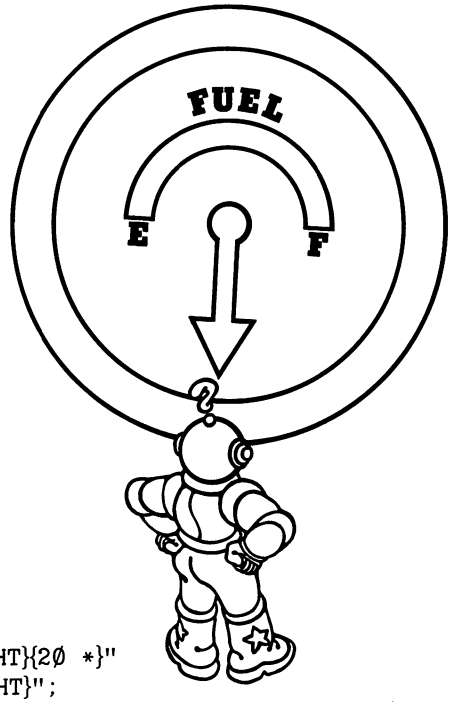
To make a game out of this, let's count how many collisions you have out of 200 times through the loop.

45 Sets both counters to 0. At the beginning of the game, there are no times (T) through the loop and no collisions (C).
55 Checks the location where you're about to go. Code 32 stands for a space. If there's something at your new location besides a space, you're about to have a collision and you are sent to the collision routine beginning at line 300.
162 Each time a meteor is displayed, adds one to the number of times through the loop.
165 When the counter gets to 200, skips to 400. (To make the game longer, choose a number larger than 200.)
400–410 Tells you how many collisions.
300–370 This is a GOSUB routine. It flashes 20 times and adds 1 to the number of collisions. After the routine is performed, the RETURN statement in line 370 returns you to the main program at line 60 (where you left off).

How Much Fuel?

Oh no! The fuel gauge is broken! Can you guess how much fuel you have left? (A full tank holds 100 gallons.)

```
5 PRINT "{CLR}"
1Ø PRINT "HOW MANY GALLONS ARE LEFT?"
2Ø F=INT(1ØØ*RND(1))
3Ø INPUT G
4Ø IF G=F THEN 1ØØ
5Ø IF G<F THEN PRINT "YOU HAVE MORE THAN THAT!"
6Ø IF G>F THEN PRINT "NOT THAT MUCH!"
7Ø GOTO 3Ø
1ØØ PRINT "RIGHT!"
```



This ending will evaluate the amount of fuel you have left. You need at least 30 gallons to land, but 60 gallons would be better.

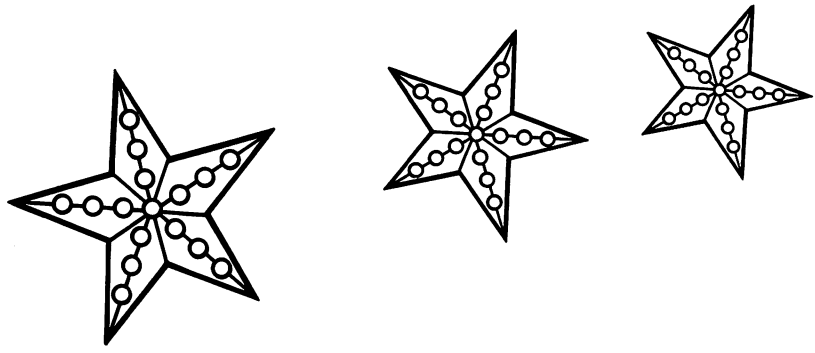
```
1ØØ PRINT "{CLR}"
11Ø PRINT "{5 DOWN}{1Ø RIGHT}{2Ø *}"
12Ø PRINT "{5 DOWN}{1Ø RIGHT}";
13Ø IF F<3Ø THEN PRINT SPC(4)"BAD NEWS!!!!":GOTO 16Ø
14Ø IF F<6Ø THEN PRINT SPC(3)"SHOULD BE OK!!":GOTO 16Ø
15Ø IF F>=6Ø THEN PRINT SPC(4)"EXCELLENT!!!"
16Ø PRINT "{5 DOWN}{1Ø RIGHT}{2Ø *}"
17Ø GOTO 17Ø
```

Main Ideas: Guessing Game
Using SPC and cursor keys to position messages

5 Clears the screen. (Press SHIFT CLR/HOME. It will look like ♥ on the screen.)
10 Asks the question.
20 Picks a number from 0 to 99 for fuel (F).
30 Allows you to input your guess.
40 Skips to line 100 if you guessed correctly.
50 If you guessed too low, prints this hint.
60 If you guessed too high, prints this hint.
70 Goes back to line 30 so you can guess again.
100 Prints "Right!"

This section centers and prints a message, depending on the amount of fuel you have.

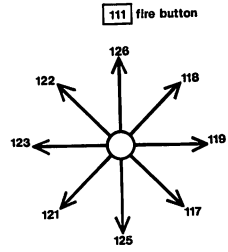
100 Clears the screen.
110 To type this line, press ↑CRSR↓ 5 times, then ←CRSR→ 10 times, and then 20 regular asterisks. This statement will move the cursor down 5 rows, and 10 spaces to the right before printing the stars. It will look like this on the screen: PRINT "QQQQQ**********"
120 Moves 5 rows down and 10 spaces to the right. The semicolon holds the cursor in position before printing the necessary message.
130–150 The SPC command is used to skip a designated number of spaces. Depending on the amount of fuel, an appropriate message is printed.
160 Moves 5 rows down, 10 spaces right and prints 20 asterisks.
170 A dummy line to freeze the image on the screen and prevent the READY signal.



Space Sketch

Insert your joystick in port 2. Before doing the program, type in this short one. It will help you to understand the main program by showing what happens when you push the joystick in different directions. Also try the fire button. Can you find all the numbers in the diagram?

```
1 J=PEEK (56320)
2 PRINT J
3 GOTO 1
```



As you near your destination, you see something awesome. The camera is broken. Sketch what you see. NASA will want to study your drawing when you return.

```
1Ø PRINT "{CLR}"
2Ø POKE 5328Ø,7
3Ø POKE 53281,Ø
4Ø L=5ØØ:C=1
5Ø POKE 1Ø24+L,16Ø
6Ø POKE 55296+L,C
7Ø J=PEEK(5632Ø)
8Ø IF J=127 THEN 7Ø
9Ø IF J=119 THEN L=L+1
1ØØ IF J=123 THEN L=L-1
11Ø IF J=126 THEN L=L-4Ø
12Ø IF J=125 THEN L=L+4Ø
19Ø IF L<Ø OR L>999 THEN 1Ø
2ØØ GOTO 5Ø
```

How about drawing diagonally, and in different colors?

```
13Ø IF J=118 THEN L=L-39
14Ø IF J=117 THEN L=L+41
15Ø IF J=122 THEN L=L-41
16Ø IF J=121 THEN L=L+39
17Ø IF J=111 THEN C=INT(16*RND(1))
```

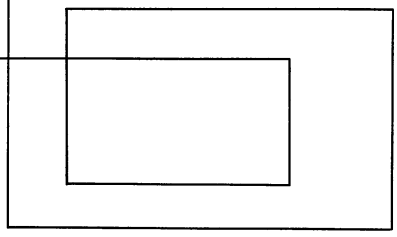
Main Idea: Using the joystick to draw

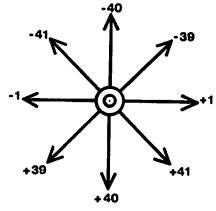
Here's how it works:

10 Clears the screen. To type the {CLR}, just press SHIFT CLR/HOME. It will look like a reversed heart on the screen.
20 Selects border color 7 (yellow).
30 Selects background color 0 (black).
40 Sets the location of the first characters at the 500th block, in the middle of the 1000-block screen. The color of the drawing will be color 1 (white).
50 Draws a white block (a reversed space) at the current location. Refer to the screen codes in the Appendix. The code for a space is 32. Add 128 to get a reversed space. Try your own combinations. For example, try code 81 (a dot) and then try 81+128, or 209, for a reversed dot.
60 Sets the color of the character being drawn.
70 Checks the joystick.
80 Goes back to 70 if the joystick is at rest and has not been pushed.
90 Adds 1 to the location if you pushed right.
100 Subtracts 1 for left.
110 Subtracts 40 (one row contains 40 positions) if you pushed up.
120 Adds 40 for down.
190 If you try to go off the screen, you must start over. Watch out for the lower right and upper left corners!
200 Goes to 50 to do the drawing.

Here are some explanations about the addition to the program:

130–160 These lines allow you to draw diagonally. When you push diagonally, you are really pushing in two directions at the same time. Refer to the diagram to help you understand where the numbers come from.
170 If the fire button is pressed, picks a new number for C, a number from 0 to 15. By changing the color you can make a multicolored drawing. You can also erase anything that you don't like by drawing over it in a color you like.

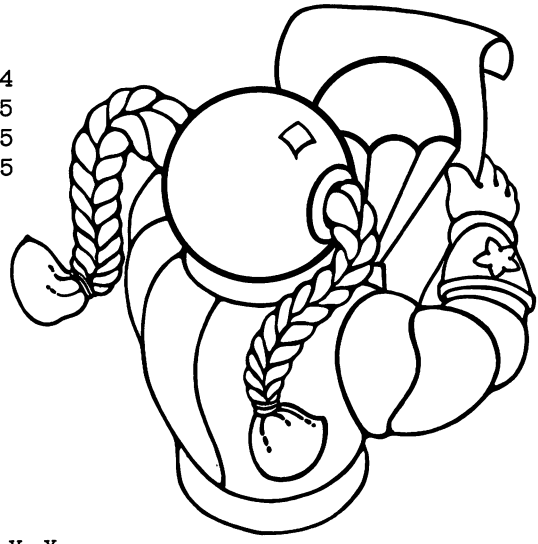




Design Your Own Parachute

Before leaving Earth, you were asked to design your own parachute. If all goes well, you will make it safely back to Earth.

```
5 PRINT "{CLR}"
1Ø POKE 5328Ø,6:POKE 53281,6
2Ø V=53248
3Ø POKE V+21,1
4Ø POKE 2Ø4Ø,13
5Ø POKE V+39,4
6Ø FOR N=Ø TO 62:READ D:POKE 832+N,D:NEXT N
1ØØ DATA 1,255,128
1Ø1 DATA 7,255,224
1Ø2 DATA 31,255,248
1Ø3 DATA 31,255,248
1Ø4 DATA 63,255,252
1Ø5 DATA 127,255,254
1Ø6 DATA 255,255,255
1Ø7 DATA 255,255,255
1Ø8 DATA 255,255,255
1Ø9 DATA 128,8,1
11Ø DATA 64,8,2
111 DATA 32,8,4
112 DATA 16,8,8
113 DATA 8,8,16
114 DATA 4,8,32
115 DATA 2,8,64
116 DATA 1,8,128
117 DATA Ø,137,Ø
118 DATA Ø,74,Ø
119 DATA Ø,6Ø,Ø
12Ø DATA Ø,6Ø,Ø
13Ø FOR X=1 TO 255:Y=X
14Ø POKE V,X:POKE V+1,Y
15Ø NEXT X
16Ø PRINT:PRINT:PRINT "WELCOME BACK TO EARTH!"
17Ø GOTO 17Ø
```

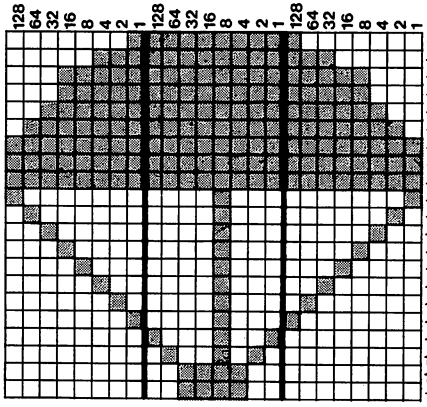


This line would make your parachute bigger:

```
55 POKE V+23,1: POKE V+29,1
```

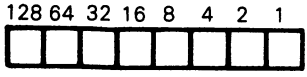
For a little excitement, add the next program to this one.

Main Ideas: Using binary numbers
Creating a sprite




```
100 DATA 1,   255,  128
101 DATA 7,   255,  224
102 DATA 31,  255,  248
103 DATA 31,  255,  248
104 DATA 63,  255,  252
105 DATA 127, 255,  254
106 DATA 255,255,  255
107 DATA 255,255,  255
108 DATA 255,255,  255
109 DATA 128, 8,    1
110 DATA 64,  8,    2
111 DATA 32.  8,    4
112 DATA 16,  8,    8
113 DATA 8,   8,    16
114 DATA 4,   8,    32
115 DATA 2,   8,    64
116 DATA 1,   8,    128
117 DATA 0,   137,  0
118 DATA 0,   74,   0
119 DATA 0,   60,   0
120 DATA 0,   60,   0
```

Read your manual for a fuller description of sprites. Depending on how many boxes you use, you can obtain any value from 0 to 255. Try a few.



5 Clears the screen. To type the {CLR} press **SHIFT** **CLR/HOME**. It will look like a reversed heart on the screen.

10 Makes a blue background and border.

20 The starting address of the video chip.

30 Enables sprite 0.

40 Sprite 0 data from the 13th block.

50 Sets the color of the parachute to purple. Refer to the Color Chart in the Appendix to select a different color.

60 Reads the data for the sprite and pokes the values into locations 832 to 894, locations reserved for sprite 0.

100–120 Data for the parachute. Refer to the diagram. Each set of three numbers stands for the boxes colored in in the first, second, and third sections of a row.

130–150 The parachute is drawn at pixels 1, 1; 2,2; 3,3; etc. Locations V (53248) and V+1 (53249) are poked again and again for the X and Y positioning of sprite 0.

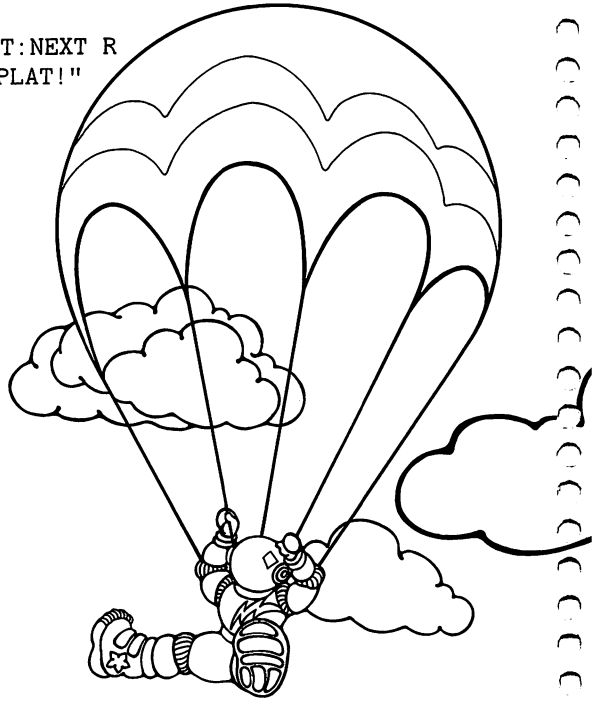
160 Prints message.

170 A dummy line used to prevent the READY signal. The line keeps going to itself, so the program "freezes." Since the program never ends, you never get a READY signal.

Happy Landing

On your descent to Earth you must decide when you need your parachute. Press "P" when you think the time is right. The timing is absolutely critical! (Add these lines to your last program.)

```
6 PRINT "PRESS P FOR PARACHUTE!"
7 GOSUB 200
200 FOR L=54272 TO 54296:POKE L,0:NEXT L
210 POKE 54296,15:POKE 54277,88:POKE 54278,89:POKE
    54275,9:POKE 54274,0
220 FOR H=250 TO 1 STEP -2
230 POKE 54276,65:POKE 54273,H:POKE 54272,100
240 FOR P=1 TO 50:NEXT P:POKE 54276,64
250 GET K$
260 IF K$="P" THEN 300
270 NEXT H
300 IF H>=100 AND H<=150 THEN RETURN
310 IF H>150 THEN PRINT "TOO EARLY! SORRY!":END
320 PRINT "TOO LATE!"
330 FOR R=1 TO 20:PRINT:NEXT R
340 PRINT "          SPLAT!"
350 GOTO 350
```

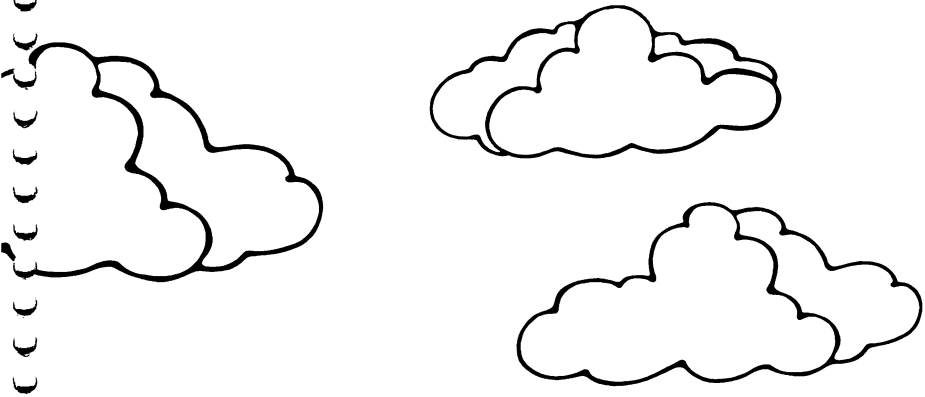


Main Ideas: Creating a sound effect by playing a sequence of notes
Using GET to interrupt a sequence

When added to the parachute program, these lines will give you a game where timing is critical.

6 Prints instructions.
7 Sends you to a GOSUB routine, a series of directions executed before returning to the main program.
200 Clears the sound locations in the computer.
210 Pokes values into locations related to creating sounds.
220–270 Plays a sequence of notes.
220 The high frequency will go from 250 to 1 by two's.
230 Turns on the waveform, pokes the high frequency value into memory and maintains 100 as the low frequency value.
240 Pauses and turns off the waveform. To slow down the sequence of notes, pick a number greater than 50. To speed up the sequence, select a number less than 50.
250 Checks to see if a key has been pressed.
260 If a P was pressed, skips to line 300.
270 Continues to play the sequence of notes.
300–350 Reacts depending on your place in the sequence of notes.
300 If you are between notes 100 and 150, returns to the program above and draws the parachute.
310 Prints this if you are between 250 and 150.
320–350 Prints "TOO LATE" and prints 20 blank rows before printing "SPLAT!" The last line is a dummy line used to prevent a READY signal.

NOTE: You may obtain the same result in line 330 by printing the ↑ CRSR ↓ key 20 times. When you use this key inside quotation marks, it prints a reversed Q on the screen: 330 PRINT "QQQQQQQQQQQQQQQQQQQQ"



Final Report

It's time to evaluate your mission. The General wants your report as soon as possible.

```
5 REM ASKS FOR INPUT
1Ø PRINT "{CLR}"
15 PRINT "YOUR NAME";
2Ø INPUT N$
25 PRINT "NAME OF GENERAL IS GEN.";
3Ø INPUT G$
35 PRINT "PLANET VISITED";
4Ø INPUT PL$
45 PRINT "LENGTH OF TRIP";
5Ø INPUT L$
55 PRINT "DAY OF WEEK";
6Ø INPUT D$
65 PRINT "A NUMBER";
7Ø INPUT N
1ØØ REM PRINTS REPORT
11Ø PRINT "{CLR}"
12Ø PRINT "REPORT OF "N$
13Ø PRINT "TO GEN."G$
14Ø PRINT
15Ø PRINT "MY MISSION TO "PL$" IS COMPLETED."
16Ø PRINT "THE MISSION LASTED "L$"."
17Ø PRINT "I WILL SEND YOU A "N" PAGE"
18Ø PRINT "REPORT NEXT "D$"."
```

Main Ideas: Inserting variables into a text
Distinguishing between numeric and string variables

5 A reminder.

10 Clears the screen. Press SHIFT CLR/HOME.

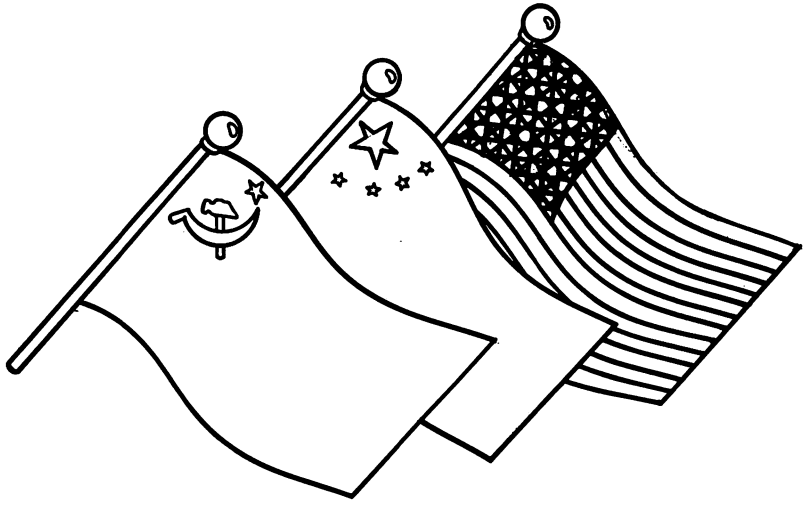
15–70 Asks you to input some variables. There are two kinds of variables. String variables (N$, G$, PL$, etc.) contain alphabetic or alphanumeric information, but numeric variables (N) can contain only numbers. When you are asked for a number (N), see what happens if you type in a letter.

100–180 Prints the report. Notice the use of quotation marks and spaces. If you want to print a space, you must include it inside the quotes. One exception: the Commodore 64 will leave a space before and after printing a number. That is why line 170 uses no extra spaces inside the quotation marks.

The Race for Space

Find out who has won the race for space.

```
1Ø PRINT "{CLR}"
2Ø PRINT:PRINT TAB(1Ø)"THE RACE FOR SPACE"
3Ø PRINT "{HOME}{7 DOWN} U.S.A."
4Ø PRINT "{HOME}{12 DOWN} U.S.S.R."
5Ø PRINT "{HOME}{17 DOWN} CHINA"
6Ø U=Ø:R=Ø:C=Ø
7Ø X=INT(3*RND(1))+1
8Ø IF U=4Ø OR R=4Ø OR C=4Ø THEN 2ØØ
9Ø IF X=1 THEN PRINT "{HOME}{8 DOWN}"SPC(U)
   "{GRN}X":U=U+1
1ØØ IF X=2 THEN PRINT "{HOME}{13 DOWN}"SPC(R)
    "{RED}*":R=R+1
11Ø IF X=3 THEN PRINT "{HOME}{18 DOWN}"SPC(C)
    "{YELO}O":C=C+1
12Ø GOTO 7Ø
2ØØ PRINT "{HOME}{21 DOWN}"
21Ø IF U<4Ø THEN PRINT "TOO BAD,CAPTAIN! YOU MUST TRY
    HARDER!"
22Ø IF U=4Ø THEN PRINT "CONGRATULATIONS,CAPTAIN!"
23Ø GOTO 23Ø
```



Main Ideas: A race
Experimenting with colors, symbols and cursor movement in the programming mode

If you haven't done so already, experiment with the keyboard in immediate mode. Type a key. Then type it again using SHIFT, and again using CTRL, and again using C=. Try each key. Whenever you need to, push RUN/STOP RESTORE to get back to the normal screen. Now you are ready to use some of these combinations in programming mode:

10 Clears the screen. Press SHIFT CLR/HOME.
20 Skips a line, counts to position 10 and prints.
30 Brings the cursor home, then down 7 rows to print "U.S.A." To type this line press CLR/HOME. Then press ↑ CRSR ↓ 7 times. On the screen, line 30 looks like this: 30 PRINT."SQQQQQQQ U.S.A."
40 Brings the cursor home, then down 12 rows to print "U.S.S.R."
50 Brings the cursor home, then down 17 rows to print "CHINA".
60 All 3 countries will begin the race at box 0.
70 Picks a number from 1 to 3.
80 Checks to see if any contestant is up to box 40. Since 39 is the last column, this would mean that someone has won the race. If there is a winner, the program skips to line 200.
90 If the computer picked a 1, the cursor goes home, then down 8 rows. It then skips the number of spaces that U has completed and prints a green clover. (To get {GRN} press CTRL 6. It looks like ↑ on the screen. When you press SHIFT X you get the ♣.) It then adds one to the number of spaces that U has travelled so that each green clover is printed *next to* the last green clover. Be sure that you do not add extra spaces.
100 If it picked a 2, the computer makes a red star for Russia. (To get {RED} press CTRL 3. It looks like £ on the screen.)
110 Similarly, a yellow circle is made for China. To get the {YELO} press CTRL 8. It looks like π on the screen. When you press SHIFT Q you will get a ●.
120 Goes back to line 70 to select the next country to move ahead.
200 Moves the cursor home, then down 21 rows.
210 If the U.S. is not the country that reached 40, it means someone else has won the race, and the computer prints this message.
220 If the U.S. is the winner, this message is printed.
230 A dummy line to freeze the message on the screen.

To run the race on a different colored screen, select a color from the color chart and add this line. I chose color 11 (gray): 15 POKE 53281, 11

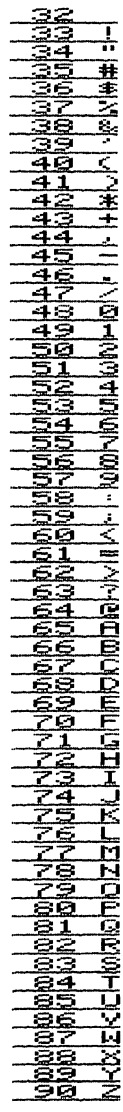
Also, experiment with the color keys and graphics symbols in lines 90, 100, and 110. Be sure to type the quotes before making changes so that the computer knows you are in the programming mode. If a line gets messed up, just type the whole line again. The new line will replace the old line.

Break the Code

The General has sent you this final message. Run the program to break the code.

```
1Ø PRINT "{CLR}{11 DOWN}"
2Ø PRINT SPC(13);
3Ø READ N
4Ø IF N=-1 THEN 1ØØ
5Ø PRINT CHR$(N);
6Ø GOTO 3Ø
7Ø DATA 79,86,69,82,32,65,78,68,
   32,79,85,84,33,-1
1ØØ GOTO 1ØØ
```

```
1Ø PRINT "{CLR}"
2Ø PRINT "WHAT CHARACTER DO YOU
   WANT TO KNOW?"
3Ø INPUT A$
4Ø A=ASC(A$)
5Ø PRINT "ITS CODE IS "A
6Ø PRINT
7Ø GOTO 2Ø
```



Main Idea: Using ASC and CHR$ to find out and use ASCII codes

The codes you see here are standard to most computers. These code numbers were used to create a secret message.

10 Clears the screen and moves the cursor down 11 rows. To get the {CLR}, press SHIFT CLR/HOME. Then press 11 ↑ CRSR ↓. On the computer screen, line 10 looks like this: 10 PRINT "♥QQQQQQQQQQ"
20 Skips 13 spaces. The semicolon keeps the cursor at the 14th position, ready to print.
30 Reads a number from the data list.
40 Checks to see if the last number has been read. If so, the program skips to line 100.
50 Prints the character itself instead of its code number. The semicolon is important because it keeps the cursor where you want it. Leave it out and see the difference!
60 Goes back to read the next number.
70 The code numbers.
100 A dummy line to freeze the message and prevent the READY signal.

To find the ASCII code for a character, say the letter M, type:

PRINT ASC ("M")

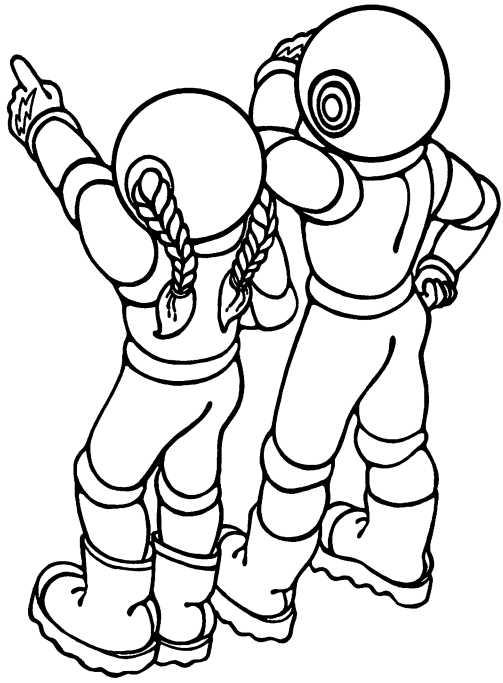
If you don't have a chart and you would like to find out codes, the program on the bottom of the page is useful for most characters:

10 Clears the screen.
20 Asks you for a character. If you type more than one character, the ASCII code of the first character will be returned.
30 Allows you to input.
40 Assigns the variable name A to the ASCII code number.
50 Prints the code number.
60 Skips a line.
70 Goes back to get the next character you want to know.

84 72 69 32 69 78 68

Appendix

Color Chart, 45
Memory Maps:
  Color Memory Map, 46
  Screen Memory Map, 47
Musical Notes, 49
Screen Display Codes, 50
Sprite Planner, 51
Typing Tips, 52

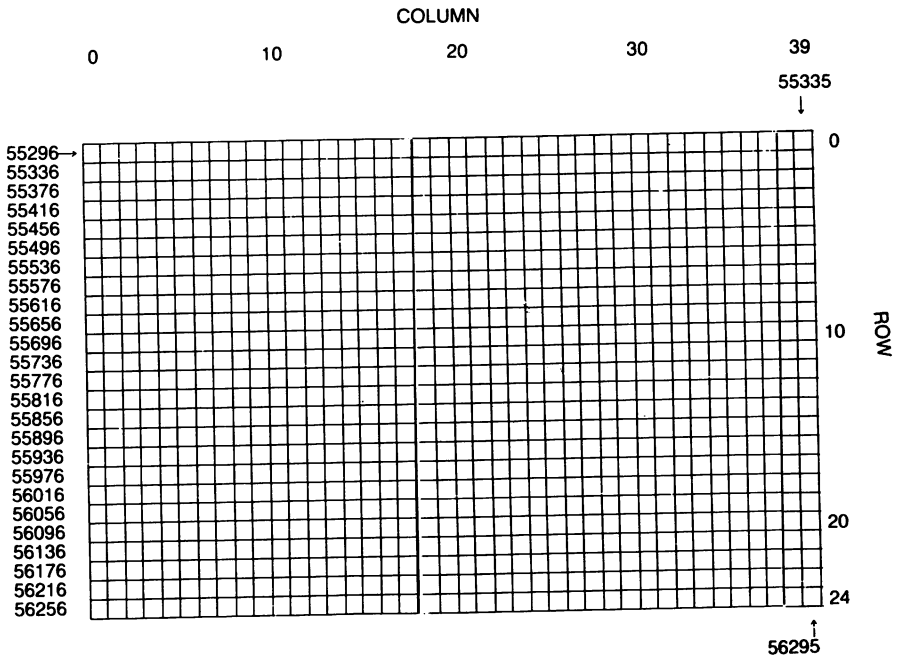


COLOR CHART

*POKE VALUE**	*COLOR*	*How it looks on the PRINTOUT*	*How it looks on the SCREEN*	*How to type it on the KEYBOARD*	*or type it using CHR$*
0	BLACK	{BLK}		CTRL 1	CHR$(144)
1	WHITE	{WHT}		CTRL 2	CHR$(5)
2	RED	{RED}		CTRL 3	CHR$(28)
3	CYAN	{CYAN}		CTRL 4	CHR$(159)
4	PURPLE	{PURP}		CTRL 5	CHR$(156)
5	GREEN	{GRN}		CTRL 6	CHR$(30)
6	BLUE	{BLUE}		CTRL 7	CHR$(31)
7	YELLOW	{YELO}		CTRL 8	CHR$(158)
8	ORANGE	{ORNG}		C= 1	CHR$(129)
9	BROWN	{BRN}		C= 2	CHR$(149)
10	LIGHT RED	{LRED}		C= 3	CHR$(150)
11	DARK GRAY	{GRY1}		C= 4	CHR$(151)
12	MEDIUM GRAY	{GRY2}		C= 5	CHR$(152)
13	LIGHT GREEN	{LGRN}		C= 6	CHR$(153)
14	LIGHT BLUE	{LBLU}		C= 7	CHR$(154)
15	LIGHT GRAY	{GRY3}		C= 8	CHR$(155)

*POKE 53281 for the background color. Example: POKE 53281,0
POKE 53280 for the border color. Example: POKE 53280,2
Refer to the Color Memory Map to POKE color at specific screen locations.

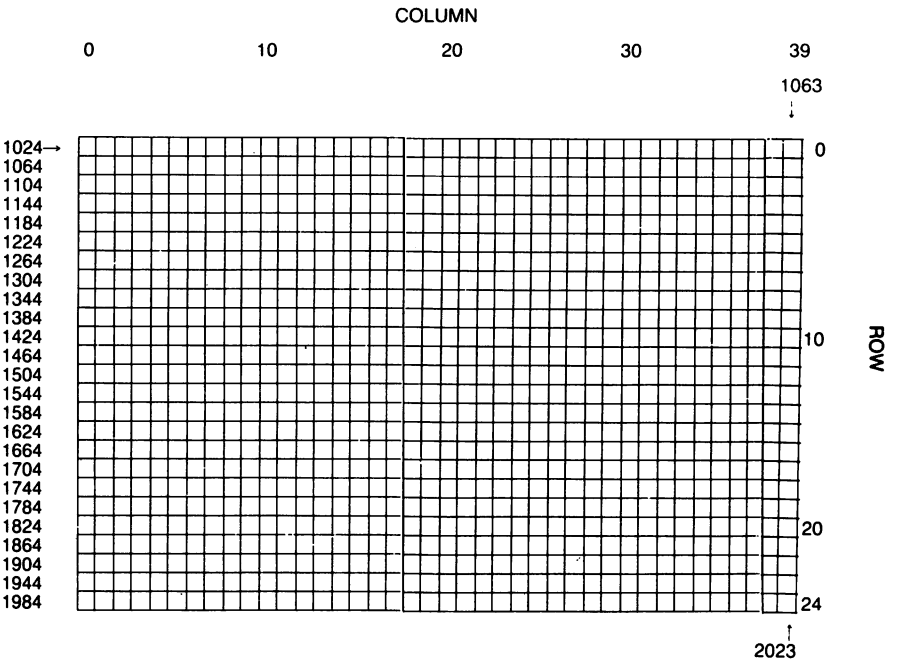
COLOR MEMORY MAP



Consult the color memory map to change the color of characters at specific screen locations. For example:

```
10 POKE 1024,1      (Makes an A in the upper left corner.)
20 POKE 55296,1     (Makes the A white.)
```

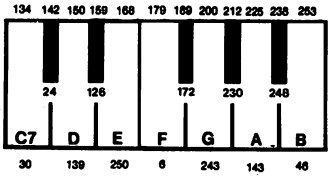
SCREEN MEMORY MAP

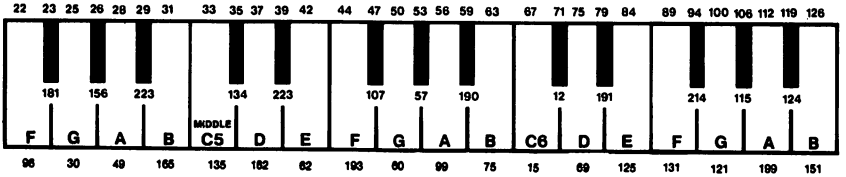


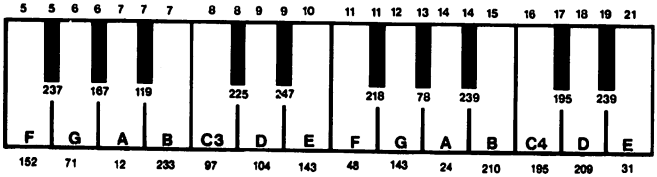
Consult the screen memory map when placing characters at specific screen locations. You will also need to refer to the chart of Screen Display Codes.

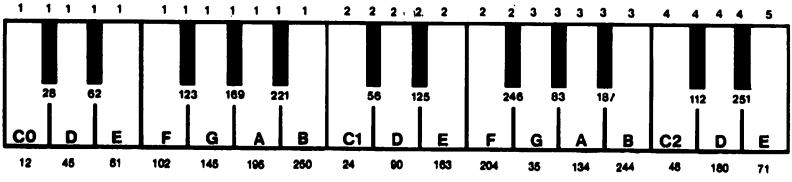
Musical Notes

There are eight octaves of notes available to you on the Commodore 64. The number above each note is the HI FREQ value. The number below each note is its LOW FREQ value.









SCREEN DISPLAY CODES

SET 1	SET 2	POKE
@		0
A	a	1
B	b	2
C	c	3
D	d	4
E	e	5
F	f	6
G	g	7
H	h	8
I	i	9
J	j	10
K	k	11
L	l	12
M	m	13
N	n	14
O	o	15
P	p	16
Q	q	17
R	r	18
S	s	19
T	t	20
U	u	21
V	v	22
W	w	23
X	x	24
Y	y	25
Z	z	26
[		27
£		28
]		29
↑		30
←		31

SET 1	SET 2	POKE
SPACE		32
!		33
"		34
#		35
$		36
%		37
&		38
'		39
(		40
)		41
*		42
+		43
,		44
-		45
.		46
/		47
0		48
1		49
2		50
3		51
4		52
5		53
6		54
7		55
8		56
9		57
:		58
;		59
<		60
=		61
>		62
?		63

SET 1	SET 2	POKE
		64
	A	65
	B	66
	C	67
	D	68
	E	69
	F	70
	G	71
	H	72
	I	73
	J	74
	K	75
	L	76
	M	77
	N	78
	O	79
	P	80
	Q	81
	R	82
	S	83
	T	84
	U	85
	V	86
	W	87
	X	88
	Y	89
	Z	90
		91
		92
		93
		94
		95

SET 1	SET 2	POKE
SPACE		96
		97
		98
		99
		100
		101
		102
		103
		104
		105
		106
		107
		108
		109
		110
		111
		112
		113
		114
		115
		116
		117
		118
		119
		120
		121
		122
		123
		124
		125
		126
		127

To POKE these characters on the screen, you need to know four things:

1. The proper screen display code number.
2. Where to put the character on the screen. (Consult the Screen Memory Map.)
3. The code number for the color. (Consult the Color Chart.)
4. Where to put the color on the screen. (Consult the Color Memory Map.)

Try this:

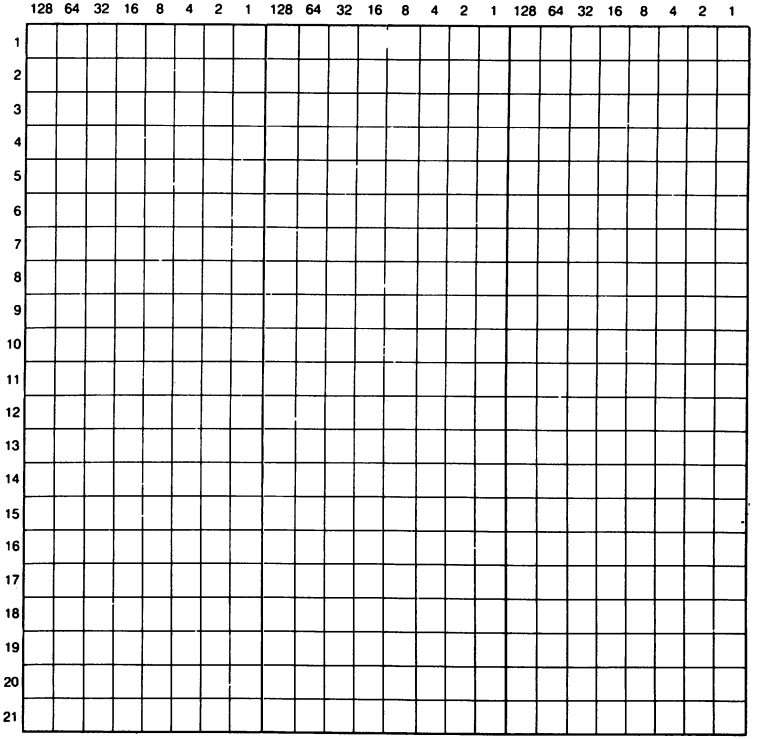
```
10 POKE 1024,1
20 POKE 55296,7
```

Did you get a yellow "A" in the upper-left corner of the screen? Now add this:

```
5 POKE 53272,22
```

Why did you get an "a"? Note: POKE 53272,22 for lower case (set 2). POKE 53272,21 to get back to upper case (set 1).

Codes from 128–255 are reversed images of codes 0–127.



SPRITE PLANNER

Use this chart to create sprites for your Commodore 64.

TYPING TIPS

How it looks on the PRINTOUT	*How it looks on the SCREEN*	*How to type it on the KEYBOARD*	*or type it using CHR$*
{CLR}	♥	SHIFT CLR/HOME	CHR$ (147)
{HOME}	S	CLR/HOME	CHR$ (19)
{DOWN}	Q	↓ CRSR ↑	CHR$ (17)
{UP}	O	SHIFT ↓ CRSR ↑	CHR$ (145)
{RIGHT}	J	→ CRSR ←	CHR$ (29)
{LEFT}	▌▌	SHIFT → CRSR ←	CHR$ (157)
{RVON}	R	CTRL 9	CHR$ (18)
{RVOF}	▬	CTRL 0	CHR$ (146)

Some keys have special graphics characters printed on them. To obtain the character on the right of the key, press SHIFT. To obtain the character on the left, press C=. Each of these characters also has a CHR$ code number that you can look up in the User's Guide that comes with the computer. On the printouts in this book, a shifted character is underlined. A character made by pressing the C= key is enclosed in special brackets { }. For example:

How it looks on the PRINTOUT	*How it looks on the SCREEN*	*How to type it on the KEYBOARD*	*or type it using CHR$*
Q	●	SHIFT Q	CHR$ (113)
{Q}	⊢	C= Q	CHR$ (171)

Refer to the Color Chart for typing information about the colors.

Note: If you find a number enclosed in the brackets, this means that you must repeatedly type the key indicated. Here are some examples:

{10 DOWN} means that you should press the ↓ CRSR ↑ key ten times.
{5 {B}} means that you should hold down the C= key and press the B five times.

Notes

This page is for you to write on.

$6.95

CLAIRE BAILEY PASSANTINO

ITTY BITTY BYTES OF SPACE

SPACE—an endless source of fascination for us all. Children, however, have the imagination to project themselves into orbit as effortlessly as they learn to talk. Now, ITTY BITTY BYTES OF SPACE encourages your child to explore the magic of space with the help of your home computer. Young readers are presented with bite-size programs and activities designed to excite their eyes, ears, and minds. Each program is accompanied by programming hints and suggestions for parents and teachers. Help your children learn, grow, and play—the computer way!

Table of Contents: A Note to Parents and Teachers • Passing the Test • Gravity • Test Your Spacecraft • Wake Up! • Kaboom! • Lift Off • Music from Mother Earth • Count the Stars • Make Your Own Kind of Music • What Is It? • Who Is It? • Hit or Miss? • Meteors • How Much Fuel? • Space Sketch • Design Your Own Parachute • Happy Landing • Final Report • The Race for Space • Break the Code

Other **Itty Bitty Bytes** Books: Matilda the Computer Cat
School Days

For more information on Creative Pastimes Books write to:
Reston Computer Group
11480 Sunset Hills Road
Reston, VA 22090

illustration by Bethann Thornburgh

0 21898 33200 4

Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia

0-8359-3320-2